UNSW College

DPST1091/CPTG1391 Introduction to Programming

Term 2 2025

COURSE OUTLINE

DIPLOMA

For detailed information about copyright, ownership and delivery of this course, see page 2.

This course is part of the following programs:

Program Name	CRICOS Course Code
*UNSW Diploma in Computer Science	102393G
^UNSW College Diploma of Computer Science	113046B

Copyright Notice

This Course Outline has been made available to you by UNSW Global Pty Limited 2025.

The material in the Course Outline may be subject to copyright under the Copyright Act 1968 (the Act).

Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.

* Delivered by UNSW College on behalf of UNSW Sydney - CRICOS Provider Code 00098G; UNSW Sydney TEQSA Provider ID: PRV12055 (Australian University)

^ Delivered by UNSW College under its own CRICOS Provider Code 01020K and TEQSA Provider ID: PRV13020 (Institute of Higher Education)

TABLE OF CONTENTS

SECTION 1

This Course Outline is designed to guide you through the course successfully. It includes information about the course, your teaching team, how to prepare for classes, details of your assessment tasks, and where to access support and assistance when needed.

Staff

Position	Name	Email
Course Convenor	Dr Pantea Aria	P.Aria@unswcollege.edu.au

Emailing Staff

When contacting staff, you must use your official UNSW College email address. In all emails, please include your:

- ZID
- First and Surname
- Course Name and Course Code

Your teachers are your first point of contact for academic advice or support with your coursework. Their email contact details are easily accessible on your online learning platform. If you have questions about your work, don't hesitate to reach out—they are available to assist you and will respond promptly.

What is expected of you

ATTENDANCE

To achieve academic success, it is recommended that you aim to achieve 100% attendance. You are expected to attend at least 80% of classes. Educational research consistently demonstrates that this level of attendance is associated with a high likelihood of achieving a passing grade.

CHECKING YOUR STUDENT EMAIL

It is important to check your student email account regularly, as all official communications, updates, and announcements related to your course will be sent there. Staying up to date with your email will ensure you do not miss any critical information or deadlines.

ONLINE LEARNING

Access to the course's online learning platform is an essential part of your studies, as all course content will be available online. You are expected to regularly log in to the platform to access lectures, assignments, resources, and updates.

YOUR CONDUCT

You are required to:

- act honestly and uphold academic integrity in all your studies
- treat all students, staff, and affiliates with courtesy and respect
- attend scheduled learning activities and engage actively in your studies
- submit your own work for all assessment tasks and avoid misconduct such as plagiarism or cheating
- use College resources, facilities, and equipment responsibly
- keep your student ID and passwords secure and confidential
- follow all policies, procedures, and lawful directions from staff
- refrain from behaviour that disrupts the teaching and learning environment or negatively impacts others
- maintain the College's reputation through responsible and respectful conduct.

What you can expect from your teachers

TEACHING SUPPORT AND RESPONSIBILITIES

The teaching team is dedicated to supporting your learning and success in this subject. You can expect the teaching team to:

- be well-prepared for each teaching session
- clearly explain course requirements and content
- provide assistance at prearranged times to address your questions or concerns
- deliver constructive and timely feedback on assessments.

PROFESSIONAL CONDUCT

The teaching team is also expected to:

- promptly report any concerns regarding academic or non-academic misconduct
- respond to student queries and concerns about the subject or relevant policies and procedures
- treat all students and staff fairly, respectfully, and in accordance with all policies and procedures.

Changes to this course as a result of student feedback

Student feedback plays a key role in improving this course and we encourage you to participate in any opportunity that allows you to provide feedback, including the My Experience surveys at the end of your teaching session. Changes may be implemented progressively to enhance your learning experience (and those of future students), such as adjustments to teaching materials, assessment tasks, or learning activities. Your input helps ensure the course remains relevant, engaging, and supportive of your academic success. Please note that all surveys are conducted anonymously and, as such, your name or student ID cannot be identified.

Policy and how it affects you

There are multiple policies that apply to students, and it is important to ensure you are referring to the correct policies for the program you are enrolled in. The relevant policies for your program, depending on your enrolment date and program type, are highlighted on the UNSW College website at: https://www.unswcollege.edu.au/about/policies/policy-documents#policies. Please review this page carefully to ensure you are accessing the appropriate policy documents, including:

- Academic Integrity Policy
- Assessment Policy
- Attendance Monitoring Policy
- Enrolment Policy
- IT Security Policy
- Sexual Misconduct Policy
- Student Health, Safety and Wellbeing Policy
- Student Grievances and Complaints Policy
- Student Misconduct Policy

Academic Integrity

Academic integrity requires you to act with honesty, trust, fairness, respect, responsibility, and courage. It is your responsibility to uphold these values by ensuring your work is your own and by appropriately acknowledging the contributions and ideas of others.

Please adhere to the UNSW College Academic Integrity Policy and Procedure (https://www.unswcollege.edu.au/about/policies/policy-documents#policies) when completing and submitting your assessments.

You are responsible for reading, understanding, and following the instructions about:

- Detecting breaches of academic integrity
- Procedural Fairness
- Investigating a breach of academic integrity
- Outcome of an investigation
- Levels of breach and penalties
- Communications and notifications about academic misconduct
- Recording breaches of academic misconduct
- Appeals

Artificial Intelligence

UNSW College has created its own Artificial Intelligence Guidelines called the 3Cs Model. You are required to adhere to these guidelines when you complete your assessments. These guidelines show you when it is appropriate to use artificial intelligence for your assignment and when it is not permitted.

Category 1: AI tools cannot be used	Category 2: AI tools can be used in assistive role	Category 3: AI tools can be used for integration
		
For this category, you need to show basic skills like remembering, understanding, and using your knowledge, which are essential for future studies and work. For example, you might need to remember and use a formula to solve a math question during a final exam or explain why a business solution is important to your colleagues. In this category, you cannot use AI tools at all. This includes in-person exams, class tests, oral exams, some labs and practicals, and discussion-based assessments. Your teacher will further explain why this category is important.	For this assessment, you may use AI tools to help you develop specific skills. For example, you might use AI to look at data, find patterns in text, and get new ideas about a topic or question. You may use AI tools to help with certain tasks in the assessment. For example, you might use AI to draft and organise content, prepare for exams, test code, translate content, give feedback, and proofread. Your teacher will explain why this category is used to make sure everyone has a fair experience. Some parts of the assessment will not allow AI.	For this assessment, you may use AI tools at different stages of your assessment. In this category, you should show your skills in using AI tools to solve problems, make decisions, and create solutions. This assessment may include a part where you must show that you can use AI in an ethical and responsible way. For example, you might use AI to generate ideas, compare content, produce summaries, analyse content, reframe content, research and find answers, and write content that other students can review and give you feedback on. Your teacher will support and guide you to ensure everyone has a fair experience.

For each assessment in your course guideline, you will see one of the category icons (Category 1, 2 or 3). You can read about the category for your assessment under the heading **Supporting Information**.

If you choose to use artificial intelligence for an assessment that prohibits the use of AI tools, you will be penalised for academic misconduct.

SECTION 2

Course Information

Course code	DPST1091/CPTG1391
Course name	Introduction to Programming
Units of credit	6
Course level	Diploma
Assumed knowledge	none

Course Summary

This course introduces students to the basics of programming. Topics covered include:

- fundamental programming concepts

- the C programming language and use of a C compiler
- programming style
- program design and organisation concepts
- program testing and debugging

The course does not assume any previous programming experience.

Course Aims

From recent innovations in AI like self-driving cars to humanoid robotics navigating complex environments, leapfrogs in battery technology to sequencing the human genome - the world is benefiting and evolving thanks to computer systems. At the core of all these systems are computers executing instructions to solve exciting problems.

In this course, you will learn the fundamentals of how we instruct computers to solve problems. You will explore the architecture and mechanics of how computers operate and how you can translate real-world problems to computer programs that solve these problems.

The concepts you learn in COMP1511/DPST1091/CPTG1391 will provide a foundation for your future endeavours in computing and, we hope, will begin to change the way you think about real-world problems.

The course aims for students to become proficient in programming using a high-level language, C. By the end of the course, students should be able to construct C programs to solve problems.

Course Learning Outcomes

At the successful completion of this course you should be able to:

1. understand the core syntax & semantics of the C programming language including types, I/O, arrays, functions, pointers, structs, file manipulation and dynamic memory allocation
2. given a problem, solve it by proficiently constructing (designing, testing, debugging) a secure, reliable, and correct C program
3. understand & employ fundamental data structures including linked lists
4. use Linux and Unix-like operating systems to develop and test software

Program Learning Outcome Alignment

Course Learning Outcome (CLO)	Program Learning Outcome (PLO)	Related Assessment
CLO1: understand the core syntax & semantics of the C programming language including types, I/O, arrays, functions, pointers, structs, file manipulation and dynamic memory allocation	PLO1: Conceptual understanding of computer underpinnings (EA1.2)	Exam, Labs, Assignments, Tutorial Performance
CLO2: given a problem, solve it by proficiently constructing (designing, testing, debugging) a secure, reliable, and correct C program	PLO2: Understanding of specialist bodies of engineering knowledge (EA1.3)	Exam, Labs, Assignments, Tutorial Performance
CLO3: understand & employ fundamental data structures including linked lists	PLO3: Understanding of underpinnings (EA1.1)	Exam, Labs, Assignments, Tutorial Performance
CLO4: use Linux and Unix-like operating systems to develop and test software	PLO4: Understanding of specialist bodies of engineering knowledge (EA1.3)	Exam, Labs, Assignments

Strategies and Approaches to Learning and Teaching

ATTENDANCE AND ENGAGEMENT REQUIREMENTS

This course will consist of 8 hours of classes each week:

- Lectures, 4 hours per week.
- Tutorials, 1 hour per week, attendance will be taken.
- Labs, 3 hours per week, attendance will be taken.

This should be supplemented by x hours of self-directed learning, which may include interacting with course content, online activities, research, and work on assessment tasks.

Face-to-face hours	Self-directed hours	Consultation hours
8	8	2

Methodology

A methodology is a roadmap for learning. The roadmap allows you to understand how the learning activities and assessments assist and support learning. The methodologies for this course include:

This course adopts a **student-centred and practice-based learning approach**, supported by **teacher-led facilitation**. The lecture component introduces key theoretical concepts using real-world case studies and practical examples to build foundational understanding. Tutorials are designed to promote **collaborative problem-solving and critical thinking**, allowing students to clarify concepts and engage actively with the material. Weekly short assessments encourage consistent engagement and reflection. Lab sessions provide hands-on experience, fostering **inquiry-based learning** through structured problem sets and optional challenge exercises that extend learning beyond the core syllabus. Peer collaboration is encouraged, and demonstrators provide ongoing support, helping students build confidence through exploration and applied practice.

METHODS

A method is the event a teacher uses to help you navigate the roadmap (methodology) used for a course. The methods for this course include:

This course is delivered through a combination of lectures, tutorials, and labs designed to support progressive and applied learning.

- **Lectures (2 hours per week):** Lectures provide a structured introduction to core programming concepts, with live demonstrations and worked examples. Students are guided through theoretical content, design principles, and code walkthroughs to build foundational understanding.
- **Tutorials (1 hour per week):** Tutorials focus on reinforcing lecture content through collaborative problem-solving. Students engage in guided discussions and short formative assessments to clarify concepts and strengthen their analytical thinking. Tutorial participation is essential and includes a performance hurdle requirement.
- **Labs (3 hours per week):** Labs offer hands-on experience with weekly programming exercises. Students practice applying concepts by building functional code, troubleshooting, and participating in mini group presentations. Lab sessions are interactive and contribute to course assessment, encouraging both independent and collaborative learning.

LEARNING AND TEACHING ACTIVITIES

A strategy is a tool or activity that teachers use during an event to support student learning.

Scaffolded Lab Exercises:

Hands-on labs where students build systems, with structured challenges and optional extension tasks.

Peer Mini-Presentations:

Short group presentations at the end of each lab to share insights and practice communication.

Formative Assessments:

Weekly labs and tutorials with individual submissions and active group participation to reinforce learning.

Tutorial Problem Solving:

Guided sessions where students collaboratively work through problems to clarify lecture material.

Pre-Class Preparation:

Lecture notes provided in advance to encourage flipped learning and better engagement during lectures.

Course Schedule and Structure

Week	Topic	Overview	Related CLO
1	Introduction to course	Introduction to course/Linux/C; data types; variables, simple I/O, expressions, If Statements	1, 4
2	More Data Types, Loops and code Style	Structs, Chars, Enums, While	1, 2, 4
3	Functions	Functions, call by value	1, 2, 4
4	One Dimensional Arrays	Arrays and Functions	1, 2, 4
5	Two Dimensional Arrays	Arrays and Functions	1, 2, 4
6	Pointers	Pointers, Functions, call by reference	1, 2, 4
7	Flexibility Week	Flexibility Week	1, 2, 4
8	Strings, Multi files projects	Debugging, Strings, Character functions and multi-file projects	1, 2, 3, 4
9	Dynamic memory allocation and extra C, Linked Lists	Dynamic memory allocation and extra C (for loops, pre/post increment), introduction to linked lists	1, 2, 3, 4
10	Linked Lists	Linked Lists and insert	1, 2, 3, 4
11	Revision	Linked lists and delete, revision	1, 2, 3, 4
12	Exam Information	Revision and exam information	1, 2, 3, 4

SECTION 3

Assessment details

This section outlines the assessment tasks for this course in detail. Use this information as a reference to guide your approach to completing each task.

Assessment Schedule

Task	Weighting	Due date	Course learning outcomes assessed
1. **Programming assignment (Looping and arrays)**	20%	Friday week 8	1, 2, 4
2. **Programming assignment (linked data structures)**	25%	Friday week 12	1, 2, 3, 4
3. **Lab exercises**	15%	Weeks 2 to 12	1, 2, 3, 4
4. **Tutorial Performance Hurdle**	0%	Weeks 2 to 12	1, 2, 3
5. **Final Exam**	40%	Exam period	1, 2, 3, 4

All marks will be determined in accordance with the UNSW College Assessment Policy and Procedure. You are strongly encouraged to attempt/submit all assessment tasks within this course.

Hurdles

Exam Hurdles:

- Hurdle requirement **#1:** in the final exam you must solve a task by writing a program that uses an **array**.
- Hurdle Requirement **#2:** in the final exam you must solve a task by writing a program that uses a **linked list**.

Tutorial Performance Hurdle:

In every tutorial from Week 2 onwards, students will be given a short question to work through and submit during their tutorial. Students must attempt and submit at least 70% of the assigned tutorial assessments to pass the course.

You **cannot** pass DPST1091/CPTG1391 unless you achieve **ALL** the above hurdles.

Assessment Guidelines

You are responsible for reading, understanding, and following the instructions about (all information below can be found in the UNSW College Assessment Procedure on this web page https://www.unswcollege.edu.au/about/policies/policy-documents#policies):

- Examinations
- Assessment tasks other than examinations
- Submission and return of assessment tasks, including rules about late assessment submissions
- Feedback on assessment
- Educational adjustments
- Special considerations including applications for extension and deferred examinations
- Supplementary assessments
- Results

ASSESSMENT WORDCOUNT

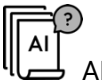
- You are required to adhere to the word count specified for each assessment.
- A 10% leeway above or below the word limit is permitted without penalty.
- Exceeding this 10% threshold will result in a penalty deduction.
- Details of word count penalties are provided in the assessment section of this Course Outline.

LATE SUBMISSION/COMPLETION OF ASSESSMENTS

Submitting assessments on time is important to ensure fairness for all students. Penalties will apply for late submissions as follows:

- A 5% penalty will be applied for each day the assessment is late, up to a maximum of five days.
- After five days, you will not be able to submit your assessment unless you have an approved educational adjustment or special consideration.
- Penalties are calculated based on the total marks available for the assessment.
- If you submit an assessment more than five days late without approved educational adjustment or special consideration, you will receive a zero mark for that assessment.

Assessment 1 – Programming assignment (Looping and arrays)

Assessment Type	Practical
Weighting	20%
Course learning outcomes addressed	CLO1: understand the core syntax & semantics of the C programming language including types, I/O, arrays, functions, pointers, structs, file manipulation and dynamic memory allocation CLO2: given a problem, solve it by proficiently constructing (designing, testing, debugging) a secure, reliable, and correct C program CLO4: use Linux and Unix-like operating systems to develop and test software
Type of collaboration	Individual
Due date and time	Week 8 Friday 9 am
Total Marks	100 marks
AI Category	 AI tools can be used in an assistive role
Submission Details	Submission through Vlab and give command

TASK OVERVIEW

task will assess your ability to use arrays, if statements, loops, and functions to solve a structured problem. The assignment builds on the foundational concepts you've learned in lectures, labs, and tutorials. You are expected to complete the work independently. Discussion with peers is limited to general programming principles, not specifics of the task. Plagiarism checks are applied to all submissions, and breaches will lead to serious academic consequences.

RATIONALE

This assignment helps you solidify your understanding of programming basics such as arrays, conditional logic, loops, and modular design through functions. You will enhance your problem-solving and code organisation skills, laying a strong foundation for more advanced topics. The task promotes independent thinking, digital literacy, and time management, all essential for further study and future work in software development.

SUPPORTING INFORMATION

In completing this assessment, you are permitted to use standard editing and referencing functions in the software you use to complete your assessment. These functions are described below. You must not use any functions that generate or paraphrase passages of text or other media, whether based on your own work or not. If your Convenor has concerns that your submission contains passages of AI-generated text or media, you may be asked to account for your work.

You are permitted to use the tools **dcc-help** and **dcc-sidekick** to help you understand the error messages you may get when compiling the code you have written.

You are permitted to use **autotest-help** to help you understand why your code may not be passing the automated tests.

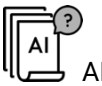
You are not permitted to submit code generated by automatic **AI tools** such as Github **Copilot**, **ChatGPT**, Google Bard in DPST1091/CPTG1391 for assignments. Submitting code generated by Github Copilot, ChatGPT, Google Bard and similar tools will be treated as plagiarism.

MARKING RUBRIC: ASSESSMENT 1 – PROGRAMMING ASSIGNMENT (LOOPING AND ARRAYS)

Performance: 80 marks out of 100	
100% for Performance	Completely Working Implementation, which exactly follows the specification
85% for Performance	Completely working implementation of Stage 1, 2 and 3.
65% for Performance	Completely working implementation of Stage 1 and Stage 2.
35% for Performance	Completely working implementation of Stage 1.

Style: 20 marks out of 100, will be allocated roughly according to the scheme provided in the assignment.

Assessment 2 – Programming assignment (linked data structures)

Assessment Type	Practical
Weighting	25%
Course learning outcomes addressed	CLO1: understand the core syntax & semantics of the C programming language including types, I/O, arrays, functions, pointers, structs, file manipulation and dynamic memory allocation CLO2: given a problem, solve it by proficiently constructing (designing, testing, debugging) a secure, reliable, and correct C program CLO3: understand & employ fundamental data structures including linked lists CLO4: use Linux and Unix-like operating systems to develop and test software
Type of collaboration	Individual
Due date and time	Week 12 Friday 9 am
Total Marks	100 marks
AI Category	 AI tools can be used in an assistive role
Submission Details	Submission through Vlab and give command

TASK OVERVIEW

In your second assignment, due in Week 12, you will develop a program that uses linked lists to manage dynamic data. You'll apply your understanding of pointers, memory allocation, and node structures to complete this task. This assignment represents a step up in complexity and reinforces key programming practices. As with the first assignment, your submission must be your own work. All code will be checked for plagiarism.

RATIONALE

This assignment develops your understanding of dynamic data structures, particularly linked lists—a core concept in computer science. You will gain hands-on experience with pointers and memory management, crucial for writing efficient and scalable programs. The task also supports critical thinking, debugging, and code documentation skills, preparing you for advanced programming tasks in academic and professional environments.

SUPPORTING INFORMATION

In completing this assessment, you are permitted to use standard editing and referencing functions in the software you use to complete your assessment. These functions are described below. You must not use any functions that generate or paraphrase passages of text or other media, whether based on your own work or not.
If your Convenor has concerns that your submission contains passages of AI-generated text or media, you may be asked to account for your work.

You are permitted to use the tools **dcc-help** and **dcc-sidekick** to help you understand the error messages you may get when compiling the code you have written.

You are permitted to use **autotest-help** to help you understand why your code may not be passing the automated tests.

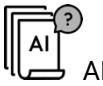
You are not permitted to submit code generated by automatic **AI tools** such as Github **Copilot**, **ChatGPT**, Google Bard in DPST1091/CPTG1391 for assignments. Submitting code generated by Github Copilot, ChatGPT, Google Bard and similar tools will be treated as plagiarism.

MARKING RUBRIC: ASSESSMENT 2 – PROGRAMMING ASSIGNMENT (LINKED LISTS)

Performance: 80 marks out of 100	
100% for Performance	Completely Working Implementation, which exactly follows the specification
85% for Performance	Completely working implementation of Stage 1, 2 and 3.
65% for Performance	Completely working implementation of Stage 1 and Stage 2.
35% for Performance	Completely working implementation of Stage 1.

Style: 20 marks out of 100, will be allocated roughly according to the scheme provided in the assignment.

Assessment 3 – Lab exercises

Assessment Type	Collaborative
Weighting	15%
Course learning outcomes addressed	CLO1: understand the core syntax & semantics of the C programming language including types, I/O, arrays, functions, pointers, structs, file manipulation and dynamic memory allocation CLO2: given a problem, solve it by proficiently constructing (designing, testing, debugging) a secure, reliable, and correct C program CLO3: understand & employ fundamental data structures including linked lists CLO4: use Linux and Unix-like operating systems to develop and test software
Type of collaboration	Individual and Group
Due date and time	Every next Monday 9 am
Total Marks	100 marks
AI Category	 AI tools can be used in an assistive role
Submission Details	Submission through Vlab and give command

TASK OVERVIEW

You will participate in weekly LabA sessions (2 hours) where you will complete programming exercises that reinforce lecture content. These labs are essential for practicing core programming skills and applying theoretical concepts in a hands-on environment. Attendance will be recorded for all lab sessions.

Each LabA includes a short **group presentation** (2–5 minutes) near the end of the session. In your group, you will briefly present one lab exercise, explain any challenges you faced (such as bugs or debugging issues), and describe your approach to solving the problem. This presentation fosters communication and collaborative skills.

You are required to **submit** your individual lab solutions via the "**give**" system by **9:00 AM on the Monday following the lab**. You may continue working on incomplete tasks after the LabA session or during LabB. Lab submissions made up to 15 minutes late will not incur a penalty. Late penalties beyond this period follow the course's lab submission policy.

Lab Exercises are capped at **15 marks**. Marks may be **deducted** from your weekly lab scores **at the end of term** if you have not actively participated in the group presentations during Lab sessions.

Optional challenge exercises (worth 0 marks) are provided for students who wish to deepen their understanding and enhance their problem-solving skills.

RATIONALE

Lab sessions give you regular practice in designing, writing, and testing code—skills that are essential for mastering programming. Group presentations help you articulate your learning, collaborate effectively, and gain confidence in communicating technical solutions. These weekly tasks also improve your time management, independence, and digital literacy. Through consistent participation, you will build a strong foundation for more advanced coursework and future employment in software development and related fields.

SUPPORTING INFORMATION

In completing this assessment, you are permitted to use standard editing and referencing functions in the software you use to complete your assessment. These functions are described below. You must not use any functions that generate or paraphrase passages of text or other media, whether based on your own work or not.
If your Convenor has concerns that your submission contains passages of AI-generated text or media, you may be asked to account for your work.

You are permitted to use the tools **dcc-help** and **dcc-sidekick** to help you understand the error messages you may get when compiling the code you have written.

You are permitted to use **autotest-help** to help you understand why your code may not be passing the automated tests.

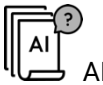
You are permitted to use **Study Buddy GPT** to help you understand why your code may not be passing the automated tests.

You are not permitted to submit code generated by any other **AI tools** such as Github **Copilot**, **ChatGPT**, Google Bard in DPST1091/CPTG1391 for assignments. Submitting code generated by Github Copilot, ChatGPT, Google Bard and similar tools will be treated as plagiarism.

MARKING RUBRIC: LAB EXERCISES (15%)

Indicator	Description	Marks
●○○	These exercises cover the most fundamental or crucial skills in the course. Every student should complete these to develop the baseline capabilities needed for the exam and future courses.	0.7
●●○	These exercises are slightly more advanced and apply core skills in more involved ways. Attempting these helps you connect lecture and tutorial concepts to practical coding tasks. Solving them supports your success in exams, assignments, and future subjects.	0.7
●●●	Designed for students seeking a challenge, these tasks apply content in more complex ways. They typically require more time and critical thinking. Useful for the later stages of assignments and more difficult final exam questions.	0
☠	These are advanced, stretch tasks that explore interesting or tough problems in Computer Science. They offer a great chance to build killer problem-solving skills but are not essential to complete the course.	0

Assessment 4 – TUTORIAL PERFORMANCE HURDLE

Assessment Type	Collaborative
Weighting	0%
Course learning outcomes addressed	CLO1: understand the core syntax & semantics of the C programming language including types, I/O, arrays, functions, pointers, structs, file manipulation and dynamic memory allocation CLO2: given a problem, solve it by proficiently constructing (designing, testing, debugging) a secure, reliable, and correct C program CLO3: understand & employ fundamental data structures including linked lists
Type of collaboration	Individual
Total Marks	0
AI Category	 AI tools can be used in an assistive role

TASK OVERVIEW

You will attend weekly tutorial sessions where you will work through examples designed to reinforce the programming concepts covered in lectures. Starting from Week 2, each tutorial will include a short-written task for you to complete and submit during the session. These tasks are designed to help you engage actively with the material and receive clarification from your tutor. Your participation and submission in at least **70% of the tutorials** is a **hurdle requirement** for passing the course. Attendance will be recorded in every tutorial session.

SUPPORTING INFORMATION

In completing this assessment, you are permitted to use standard editing and referencing functions in the software you use to complete your assessment. These functions are described below. You must not use any functions that generate or paraphrase passages of text or other media, whether based on your own work or not. If your Convenor has concerns that your submission contains passages of AI-generated text or media, you may be asked to account for your work.

You are permitted to use the tools **dcc-help** and **dcc-sidekick** to help you understand the error messages you may get when compiling the code you have written.

You are permitted to use **autotest-help** to help you understand why your code may not be passing the automated tests.

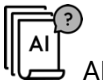
You are permitted to use **Study Buddy GPT** to help you understand why your code may not be passing the automated tests.

You are not permitted to submit code generated by any other **AI tools** such as Github **Copilot**, **ChatGPT**, Google Bard in DPST1091/CPTG1391 for assignments. Submitting code generated by Github Copilot, ChatGPT, Google Bard and similar tools will be treated as plagiarism.

MARKING RUBRIC: TUTORIAL PERFORMANCE HURDLE (0%)

Criteria	Description	Student Guidance
Participation ≥ 70%	You have attended tutorials and submitted at least 70% of the weekly tutorial tasks (starting from Week 2).	Actively participate in tutorials, complete the weekly tasks, and hand it to the tutor during the session to meet the hurdle requirement.
No or Minimal Participation	You missed most tutorials and submitted fewer than 70% of the required tutorial tasks.	You did not meet the hurdle requirement. This results in course failure, regardless of marks in other assessments.

Assessment 5 – FINAL EXAM

Assessment Type	Practical
Weighting	40%
Course learning outcomes addressed	CLO1: understand the core syntax & semantics of the C programming language including types, I/O, arrays, functions, pointers, structs, file manipulation and dynamic memory allocation CLO2: given a problem, solve it by proficiently constructing (designing, testing, debugging) a secure, reliable, and correct C program CLO3: understand & employ fundamental data structures including linked lists CLO4: use Linux and Unix-like operating systems to develop and test software
Type of collaboration	Individual
Total Marks	100
AI Category	 AI tools can be used in an assistive role

TASK OVERVIEW

The format and mode of the exam will be advised closer to the end of the term. It will contain implementation tasks that will require you to write C programs. It will also contain sections which require you to read code or answer questions to show your knowledge of programming. During this exam you will be able to execute, debug and test your answers. The implementation tasks will be like those encountered in your weekly labs.

We will provide you with sample questions in the last week of the course.

This assessment ensures that you engage consistently with the course content through active participation and guided problem-solving. By completing tutorial tasks each week, you will build your skills incrementally, clarify difficult concepts, and improve your confidence in applying theoretical knowledge. The hurdle promotes regular study habits and collaborative learning—essential qualities for success in both academic and professional environments. Meeting this requirement is necessary to progress in the course and future programming subjects.

SUPPORTING INFORMATION

In completing this assessment, you are permitted to use standard editing and referencing functions in the software you use to complete your assessment. These functions are described below. You must not use any functions that generate or paraphrase passages of text or other media, whether based on your own work or not.
If your Convenor has concerns that your submission contains passages of AI-generated text or media, you may be asked to account for your work.

You are permitted to use the tools **dcc-help** and **dcc-sidekick** to help you understand the error messages you may get when compiling the code you have written.

You are permitted to use **autotest-help** to help you understand why your code may not be passing the automated tests.

You are not permitted to submit code generated by any other **AI tools** such as Github **Copilot**, **ChatGPT**, Google Bard in DPST1091/CPTG1391 for assignments. Submitting code generated by Github Copilot, ChatGPT, Google Bard and similar tools will be treated as plagiarism.

Final Mark

Your final mark for this course will be computed using the above assessments as follows:

Component	Formula / Logic	Notes
CourseWorkMark	LabMark + Ass1Mark + Ass2Mark	Out of **60**
ExamMark	ExamPracMark + ExamTheoryMark	Out of **40**
HurdlesOK	Pass(Exam List Hurdle & Exam Array Hurdle & Tutorial Performance Hurdle)	Result: **True/False**
FinalMark	CourseWorkMark + ExamMark	Out of **100**
FinalGrade	- UF if !HurdlesOK && FinalMark ≥ 50 - FL if FinalMark < 50 - PS if 50 ≤ FinalMark < 65 - CR if 65 ≤ FinalMark < 75 - DN if 75 ≤ FinalMark < 85 - HD if FinalMark ≥ 85	UF = **Unsatisfactory Fail** FL = **Fail** PS = **Pass** CR = **Credit** DN = **Distinction** HD = **High Distinction**

SECTION 4

Essential Resources for Students

Essential resources are the materials and technologies you will require to complete this course. The essential resources for this course include:

1. Pen and paper
2. Microsoft Office 365 suite as provided by the College
3. A Windows or Apple Mac laptop (Electronic Device Requirements)
4. An approved calculator (Exam Approved Calculators and Computers) (change accordingly or remove based on department .e.g., fx-82AU)

Support for Students

UNSW College offers a range of support services to help you with your studies and student experience:

Resources	
UNSW College Current Students Website	A wide range of resources are available to support your academic and personal success. These include study advice, time management tools, course planning assistance, and academic support services. The Current Student Website also provides quick links to essential information, such as: • Health and Wellbeing resources • Academic support and learning tools • Student news and events • Forms and key administrative services Visit the Current Student Website at https://my.unswcollege.edu.au/ to access everything you need in one place.
Teacher Consultations	Schedule time with your teachers for study advice and subject-related support.
Study Club	Book a place through your online learning platform under Student Consultations to join group study sessions.
Student Advisers	Located at the Student Services Office (Level 1, Building L5), Student Advisers can assist with time management, study skills, course planning, and personal issues affecting your studies. Book appointments online via the Support section of the UNSW College Student Hub.
General Assistance	For help with matters such as accommodation, student activities, or personal concerns, visit the Student Services Office.
Subject Consultations	Times and locations will be provided by your teachers or can be found on the online learning platform.

Take advantage of these services to support your learning and overall success at UNSW College.

References

ESSENTIAL TEXTS

The optional textbook for the course is: Programming, Problem Solving, and Abstraction with C by Alistair Moffat , ISBN 978 1 74103 080 3, which can be purchased from the UNSW Bookshop.

Last amended:	May, 2025

ACKNOWLEDGEMENTS

Contributors: Pantea Aria, Andrew Taylor